Oracle® Cloud Native Environment

Concepts for Release 1.2

F40895-03
November 2021

Oracle Legal Notices

Table of Contents

Preface

This document provides an overview of the different components of Oracle Cloud Native Environment and explains key concepts that are essential to working with Oracle Cloud Native Environment.

Document generated on: 2021-11-17 (revision: 1158)

Audience

This document is written for system administrators and developers who want to learn the fundamentals of Oracle Cloud Native Environment and require a conceptual overview of the different elements that make up and are used with in an Oracle Cloud Native Environment. It is assumed that readers have a general understanding of the Oracle Linux operating system and container concepts.

Related Documents

The latest version of this document and other documentation for this product are available at:

https://docs.oracle.com/en/operating-systems/olcne/

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Chapter 1 Introduction

Oracle Cloud Native Environment is a fully integrated suite for the development and management of cloud native applications.

Oracle Cloud Native Environment is a curated set of open source projects that are based on open standards, specifications and APIs defined by the Open Container Initiative (OCI) and Cloud Native Computing Foundation (CNCF) that can be easily deployed, have been tested for interoperability and for which enterprise-grade support is offered. Oracle Cloud Native Environment delivers a simplified framework for installations, updates, upgrades and configuration of key features for orchestrating microservices.

Oracle Cloud Native Environment uses Kubernetes to deploy and manage containers. When you create an environment, in addition to Kubernetes nodes, the Oracle Cloud Native Environment Platform API Server must be installed on a server, and is needed to perform a deployment and manage modules. The term *module* refers to a packaged software component that can be deployed to provide both core and optional cluster-wide functionality. The Kubernetes module for Oracle Cloud Native Environment is the core module, and automatically installs and configures Kubernetes, CRI-O, runC and Kata Containers on the Kubernetes nodes and brings up a Kubernetes cluster.

The Kubernetes nodes run an Oracle Cloud Native Environment Platform Agent. The Platform Agent communicates with the Platform API Server to manage the deployment of modules.

The Oracle Cloud Native Environment Platform Command-Line Interface performs the validation and deployment of modules to the nodes, enabling easy deployment of modules such as the Kubernetes module. The required software for modules is configured by the Platform CLI, such as Kubernetes, CRI-O, runC, Kata Containers, CoreDNS and Flannel.

An optional module is the Istio module for Oracle Cloud Native Environment which is used to deploy a service mesh on top of the Kubernetes cluster. The Istio module requires Helm, Prometheus and Grafana, and these are also deployed along with Istio.

Chapter 2 Components

This chapter contains information on the components that are used to create Oracle Cloud Native Environment.

2.1 Container Runtimes

Containers are the fundamental infrastructure to deploy modern cloud applications. Oracle delivers the tools to create and provision Open Container Initiative (OCI)-compliant containers using CRI-O.

CRI-O, an implementation of the Kubernetes CRI (Container Runtime Interface) to enable using Open Container Initiative compatible runtimes, is included with Oracle Cloud Native Environment. CRI-O allows you to run either runC or Kata Containers containers directly from Kubernetes, without any unnecessary code or tooling.

2.2 Container Orchestration

The version of Kubernetes used in Oracle Cloud Native Environment is based on the upstream Kubernetes project, and released under the CNCF Kubernetes Certified Conformance program. The Platform API Server simplifies the configuration and set up of the Kubernetes module to create a Kubernetes cluster, with support for backup and recovery. The Kubernetes module is developed for Oracle Linux and integrates with CRI-O to provide a comprehensive container and orchestration environment for the delivery of microservices and next-generation application development.

2.3 Cloud Native Networking

The CNCF Flannel project provides the overlay network used by Kubernetes, and simplifies container-to-container networking.

The Container Network Interface (CNI) project, currently incubating under CNCF, seeks to simplify networking for container workloads by defining a common network interface for containers. The CNI plug-in is included with Oracle Cloud Native Environment.

2.4 Cloud Native Storage

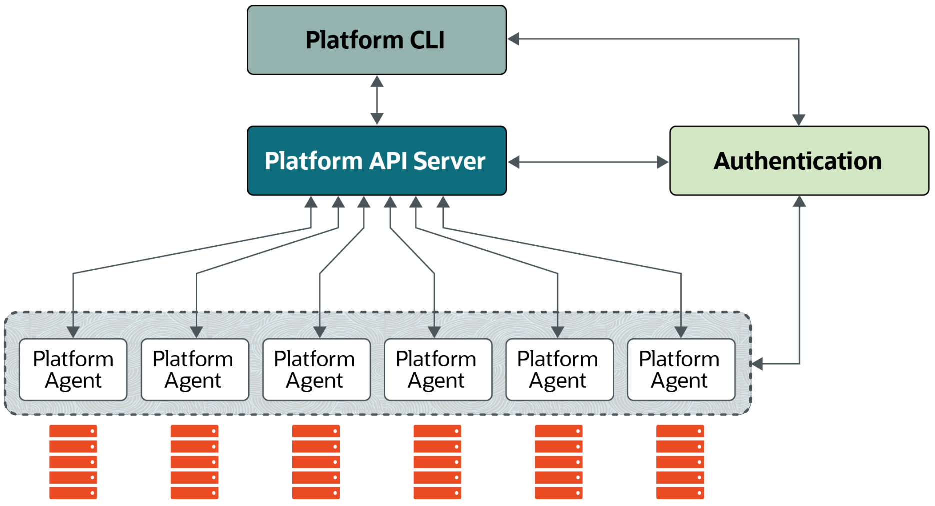
There are a number of storage projects associated with the CNCF foundation, and several providers are included by default in Kubernetes, including a plug-in for Gluster Storage for Oracle Linux.

Storage integration is provided through the use of plug-ins, referred to as the Container Storage Interface (CSI). The plug-ins adhere to a standard specification.

Chapter 3 Architecture

Oracle Cloud Native Environment is built from a number of discrete components. You interact with the environment directly using the Platform CLI. The Platform API Server interacts with the Platform Agent on each Kubernetes node. The Platform Agent is responsible for handling host-level operations on behalf of the Platform API Server.

Figure 3.1 Architecture



3.1 Platform API Server

The Platform API Server performs the business logic and manages all entities, from hosts to microservices. The Platform API Server is responsible for managing the state of the environment, including the deployment and configuration of modules to one or more nodes in a cluster.

3.2 Platform Agent

The Platform Agent runs on each host to proxy requests from the Platform API Server to small worker applications. The primary reason for this is to make sure the Platform Agent process uses as little memory as possible. The Platform Agent refers to the union of the Platform Agent process and associated worker applications.

The Platform Agent knows how to gather the state of resources on its host and to change the state of those resources. That is, the Platform Agent knows if a firewall port is open or closed, or if a package is installed and at which version. It also knows how to close that port if it is open, upgrade the package if it is old, or install the package if it is not installed.

3.3 Platform CLI

The Platform CLI is used to communicate with the Platform API Server. The Platform CLI is a simple application (the `olcnectl` command) that converts the input to Platform API Server calls. No business logic takes place in the Platform CLI. Parsing of the commands entered into the Platform CLI takes place in the Platform API Server.

The Platform CLI must be installed on an *operator* node. The operator node is used to deploy an environment.

3.4 Authentication

Standard X.509 certificates are used to establish node identity and authentication. The Platform API Server, Platform CLI and the Platform Agent on Kubernetes nodes require a valid certificate chain that allow each component to mutually authenticate. Without these certificates, connections between the components and nodes are rejected.

X.509 certificates can be created and distributed manually, or using an authentication server such as Vault by HashiCorp.

Chapter 4 Environments and Modules

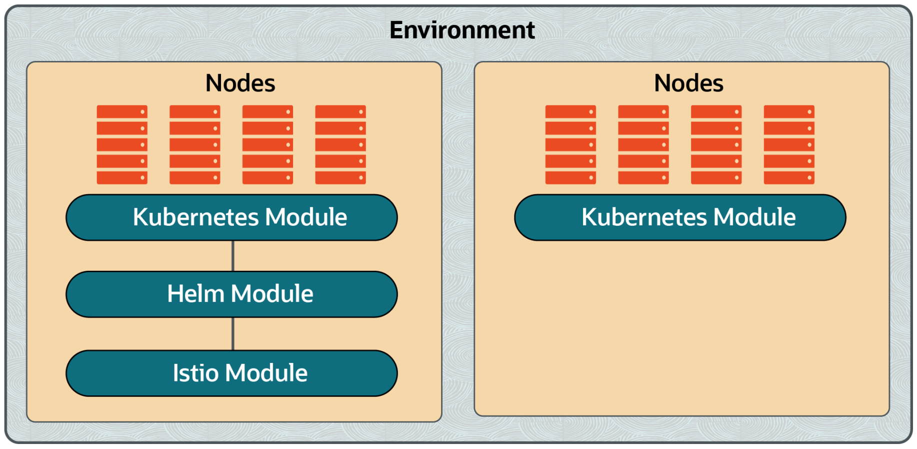
This chapter introduces the concepts of environments and modules in Oracle Cloud Native Environment.

4.1 Environments

An *environment* is a namespace that encapsulates the software installed and managed by Oracle Cloud Native Environment. Each environment contains at least the Kubernetes module.

The Platform CLI allows you to create and manage multiple deployments. Each deployment contains an environment, and each environment may potentially contain multiple modules. This allows you to create multiple Kubernetes clusters using the same Oracle Cloud Native Environment installation. Each Kubernetes cluster must have dedicated nodes, that is, a server cannot be used in two clusters, or environments.

Figure 4.1 Environments



4.2 Modules

A *module* is a curated unit of software that can be installed and managed by Oracle Cloud Native Environment. A module fulfills at least one specific role in a deployment. Modules that fulfill the same roles may be swapped out in a managed way. Modules may encapsulate other modules.

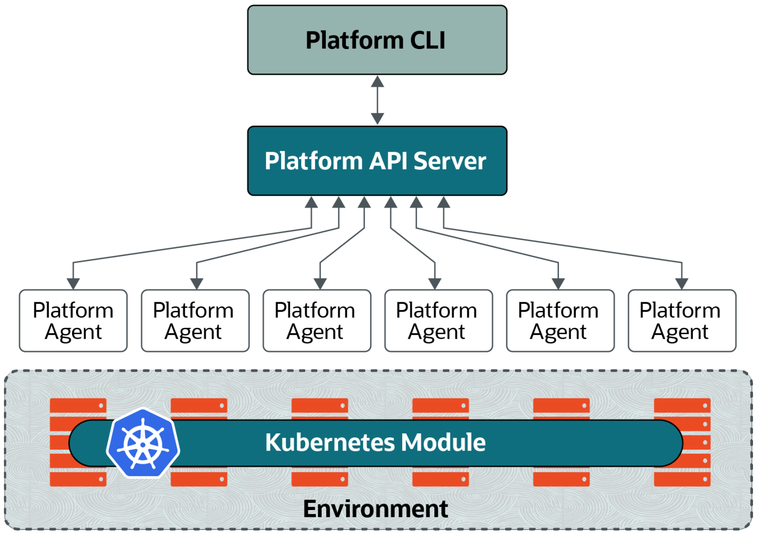
The available modules are:

- Kubernetes
- Istio
- Helm
- Prometheus

4.2.1 Kubernetes Module

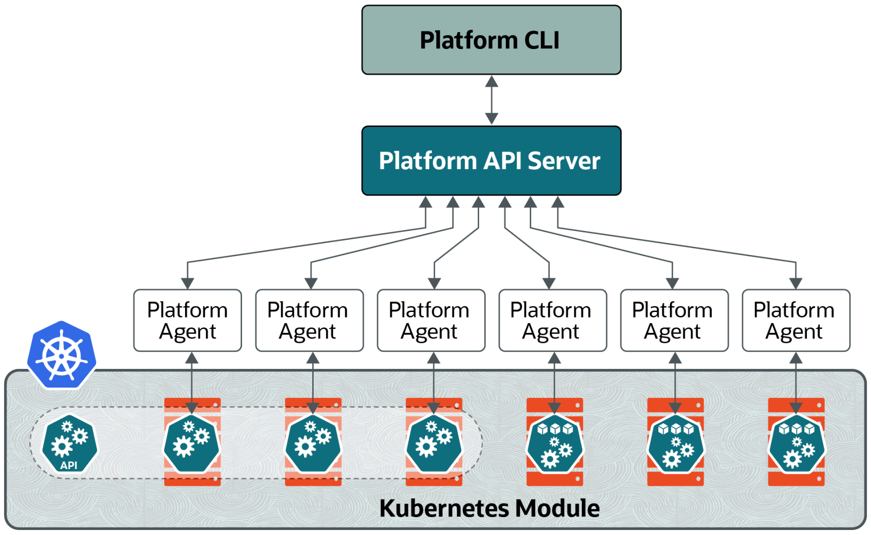
The core module in Oracle Cloud Native Environment is the Kubernetes module. The Kubernetes module is used to deploy a Kubernetes cluster in an environment.

Figure 4.2 Kubernetes module



The Kubernetes module installs and configures Kubernetes on the nodes and sets up the cluster.

Figure 4.3 Kubernetes cluster



The Kubernetes module includes sub-components, such as:

- **Flannel**: The default overlay network for a Kubernetes cluster.
- **CoreDNS**: The DNS server for a Kubernetes cluster.
- **CRI-O**: Manages the container runtime for a Kubernetes cluster.
- **runC**: The default container runtime.
- **Kata Containers**: An optional lightweight virtual machine runtime.

For more information about installing and using the Kubernetes module, see *Container Orchestration*.

4.2.2 Istio Module

Istio is a fully featured service mesh for microservices in Kubernetes clusters. Istio can handle most aspects of microservice management, for example, identity, authentication, transport security, metric scraping, and so on.

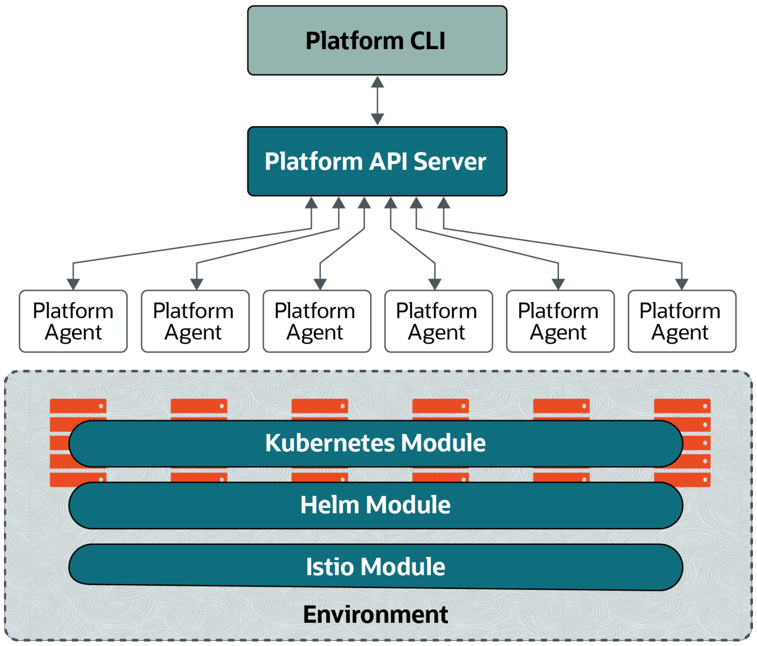
The Istio module for Oracle Cloud Native Environment installs Istio into a Kubernetes module (cluster), and uses a Helm module to perform the deployment.

The Istio module installs a number of components that are used exclusively by Istio:

- Egress gateway
- Ingress gateway
- Pilot
- Grafana
- Prometheus

When you deploy the Istio module, an embedded instance of Prometheus is also deployed. Prometheus is used to monitor and gather metrics about the Kubernetes cluster. Another embedded component that is deployed is Grafana. Grafana is a monitoring and visualization tool for time-series data stored in a database which in this case is Prometheus. Grafana enables you to visually query and monitor the network traffic and services in your Kubernetes cluster. Grafana includes browser-based dashboards to visualize the cluster metrics gathered from Prometheus.

Figure 4.4 Istio module



For more information about installing and using the Istio module, see *Service Mesh*.

4.2.3 Helm Module

Helm is a package manager for Kubernetes. Helm simplifies the task of deploying and managing software inside Kubernetes clusters. Helm uses *charts* to manage the packages that it can deploy.

The Helm module for Oracle Cloud Native Environment installs Helm into a Kubernetes module (cluster). The Helm module is used by the Platform API Server to install additional modules such as the Istio module.

Note

In this release, the Helm module should only be used in the context of an Istio module deployment.

4.2.4 Prometheus Module

Prometheus is a systems monitoring and alerting toolkit that collects and stores metrics and other time series data from various sources and presents it in an easily retrievable manner.

The Prometheus module for Oracle Cloud Native Environment is pre-configured with rich monitoring of important systems inside a Kubernetes cluster.

The Prometheus module is deployed by the Helm module into a Kubernetes cluster. The Prometheus module is required by the Istio module and is used to create an embedded instance of Prometheus for use by Istio.

Note

In this release, the Prometheus module should only be used in the context of an Istio module deployment.

Chapter 5 Network Planes

This chapter contains information about the Oracle Cloud Native Environment management, control and data planes.

5.1 Management Plane

The management plane consists of the components that make up the Oracle Cloud Native Environment platform, that is, the Platform API Server, the Platform Agent, and the Platform CLI.

Communication between the components is secured using Transport Layer Security (TLS). You can configure the cipher suites to use for TLS for the management plane.

You can set up the X.509 certificates used for TLS before you create environment, or have a certificate management application, such as Vault, manage these for you.

5.2 Control Plane

The control plane contains the Kubernetes components and any load balancer.

Kubernetes has a sophisticated networking model with many options that allow users to finely tune the networking configuration. Oracle Cloud Native Environment simplifies the Kubernetes networking by setting network defaults that align closely with community best practices.

By default, all Kubernetes services are bound to the network interface that handles the default route for the system. The default route is set to the network interface used by the Platform Agent, and it is used for both the Kubernetes control plane and the data plane.

There are two motivations behind this choice. The first is that the Platform API Server always needs to be able to communicate with the Kubernetes API server. By making sure the Kubernetes API server is bound to the same interface as the Platform Agent, this condition is always met. Also, if nodes have multiple network interfaces, it will usually be the case that the sensitive networks are not the networks that Oracle Cloud Native Environment is using to communicate.

When deploying a highly available cluster having multiple control plane nodes with an internal load balancer, the Platform API Server uses the same network interface as was set to host the Kubernetes control plane services to host the virtual IP address.

5.3 Data Plane

The data plane is the network used by the pods running on Kubernetes.

The same algorithm for determining the default control plane interface is used when instantiating the Kubernetes pod network. That is, the network interface used by the Platform Agent is used for both the Kubernetes control plane and the data plane. In multi-network environments, this may not be the best choice. Oracle Cloud Native Environment allows you to customize the network interface used for pod networking when you create the Kubernetes module. When Flannel is brought up, it uses the network interface you specify for the pod network.

Chapter 6 Highly Available Clusters

This chapter contains high level information about the types of highly available (HA) Kubernetes clusters you can deploy using Oracle Cloud Native Environment.

Kubernetes can be deployed with more than one replica of the control plane node. Automated failover to those replicas provides a more scalable and resilient service. This type of cluster deployment is referred to in this document as an HA cluster.

Important

To create an HA cluster you need at least three control plane nodes and two worker nodes.

Creating an HA cluster with three control plane nodes ensures replication of configuration data between them through the distributed key store, `etcd`, so your HA cluster is resilient to a single control plane node failing without any loss of data or uptime. If more than one control plane node fails, you should restore the control plane nodes in the cluster from a backup file to avoid data loss.

Oracle Cloud Native Environment implements the Kubernetes stacked `etcd` topology, where `etcd` runs on the control plane nodes. For more information on this topology, see the upstream documentation at:

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/ha-topology/#stacked-etcd-topology

6.1 Load Balancer

An HA cluster needs a load balancer to provide high availability of the control plane nodes. A load balancer communicates with the Kubernetes API server to maintain high availability of the control plane nodes.

You can use your own load balancer instance, or have the Platform CLI install a load balancer on the control plane nodes.

For more information about the configuration requirements of the load balancer, see *Getting Started*.

6.2 Highly Available Cluster with External Load Balancer

When you set up an HA cluster to use an external load balancer, the load balancer is used to manage the resource availability and efficiency of your control plane nodes to make sure instances of the Kubernetes API server on control plane nodes can fail without impacting cluster availability.

If you want to use your own load balancer implementation, it should be set up and ready to use before you perform an HA cluster deployment. The load balancer hostname and port is entered as an option when you create the Kubernetes module.

6.3 Highly Available Cluster with Internal Load Balancer

When you set up an HA cluster to use an internal load balancer, the Platform CLI installs NGINX and Keepalived on the control plane nodes to enable the container-based deployment of the load balancer. The internal load balancer configures the native active-active high availability solution for the Kubernetes API server.

NGINX improves the resource availability and efficiency of your control plane nodes to make sure instances of the Kubernetes API server on control plane nodes can fail without impacting cluster availability.

If you use the internal load balancer, you must set aside an IP address on the control plane network to use as a virtual IP address. The Keepalived instance makes sure the virtual IP address is always reachable by monitoring the health of other control plane nodes and appropriating the IP address if a node fails. Keepalived is used to failover automatically to a standby control plane node if problems occur.

As part of deploying the internal load balancer, the `olcne-nginx` and `keepalived` services are enabled and started on the control plane nodes.

Terminology

Environment
: A namespace that encapsulates the software installed and managed by Oracle Cloud Native Environment. Each environment contains at least the Kubernetes module.

Kubernetes Node
: A host in an Oracle Cloud Native Environment that contains the Platform Agent and other required software to run as a Kubernetes cluster member, either as a control plane or worker node.

Module
: A module represents some installable object in an environment. The most common module is the Kubernetes module, as it is required for most other modules. A module may encapsulate other modules.

Node
: A host system that is a member of an Oracle Cloud Native Environment, including hosts used as Kubernetes control plane and worker nodes, and hosts that run the Platform API Server and Platform CLI.

Operator Node
: A host in an Oracle Cloud Native Environment that contains the Platform CLI. This node may also be a Platform API Server node.

Platform Agent
: The Oracle Cloud Native Environment Platform Agent is a software agent installed on all nodes which is used by the Platform API Server to report and change state as directed by the Platform API Server.

Platform API Server
: The Oracle Cloud Native Environment Platform API Server manages the state of one or more environments, including the deployment and configuration of modules to one or more nodes in a cluster.

Platform API Server Node
: A host in an Oracle Cloud Native Environment that contains the Platform API Server. This node may also be an operator node.

Platform CLI
: The Oracle Cloud Native Environment Platform Command-Line Interface used to create and manage deployments. The `olcnectl` command.